Programming Examples

Example-2:

Banker's Algorithm
Level : Medium

Banker's Algorithm- Program Problem Statement

- The Banker's algorithm is an algorithm developed for resource allocation and to avoid deadlocks. The Banker's algorithm is run on the operating system to determine whether a state is safe or not. System can have multiple resources. Only if a state is safe then deadlock would not happen. For the Banker's algorithm to work, it needs to know three things: The algorithm works by checking whether all the currently running processes can acquire their marked maximum resources from the currently available resources.

  The System processes will run in a sequential manner only. Only after finishing a process, other processes will be started. After finishing a process, the process will release all his used resources. If any of the process fails to achieve its maximum resources the state would be considered as unsafe.

  In this problem you need to implement getState() method, which will return 0 in case the state is safe otherwise in case of the state is unsafe , it will return first process number which fails to get maximum resources.

The prototype of the function is:

- **public int getState(int allocated_res[][], int max_res[][], int avl1[])**

- where **allocated_res**, **max_res** and **avl1** are the matrices defining the currently allocated resources, maximum resources and available resources respectively.

Constraint

- The number of resources must not be greater than 5.
- Example 1
  - Input
  - Available system resources(avl_res[]):
    A B C D
    3 1 1 2
- Processes (currently allocated resources):
  ABCD
- P1 1221
- P2 1033
- P3 1110
- Processes (maximum resources):
  - ABCD
  - P1 3322
  - P2 1234
  - P3 1150
- Output Function "getState" returns 0
- **Explanation**
  1. P1 acquires 2 A, 1 B and 1 D more resources to achieve its maximum. Since the available system resources still has 1 A, no B, 1 C and 1 D resource available
  2. P1 terminates, returning 3 A, 3 B, 2 C and 2 D resources to the system. The system now has 4 A, 3 B, 3 C and 3 D resources available.
  3. P2 acquires 2 B and 1 D extra resources, then terminates, returning all its resources. The system now has 5 A, 3 B, 6 C and 6 D resources.
  4. P3 acquires 4 C resources and terminates. The system now has all resources: 6 A, 4 B, 7 C and 6 D.
  5. Because all processes were able to terminate, this state is safe.

- Example 2
- Input   Available system resources:
  A B C D
  3 0 1 2
  Processes (currently allocated resources):
  ABCD
- P1 1221
- P2 1133
- P3 1110
- Processes (maximum resources):
- ABCD
- P1 3322
- P2 1234
- P3 1150

- Output
  - Function "getState" returns 1
- **Explanation**
  P1 is unable to acquire enough B resources.
  So the process 1 cannot acquire enough resources, so this state is not safe.
  The method getState returns the process number of the first process i.e. process 1 that fails to achieve its maximum resource.
- For Java solutions
  - Package Name :test.bankersalgo
  - File Name :BankersAlgo.java
  - Class Name :BankersAlgo
  - Function Name :public int getState(int allocated_res[][], int max_res[][], int avl1[])

- **General Instructions** The package names, class names, method signatures to be used are mentioned in the problem statement. Do not use your own names or change the method signatures and fields. You can add any number of additional methods.

Pseudo Code

Bankers Algorithm

- Check whether the number of resources is greater than 5 or not. If it is greater, then return 0.
- Check how much more of each resource, each processes need to gain the maximum.
- Run a loop to check whether the need for each resource is greater than the available resources for all the processes.
- If it is greater for any of the processes then return the process number.
- Run a loop to add the allocated resources to the available resources for each process.

Solution

```
package test.bankersalgo;

 public class BankersAlgo {

    public int getState( int allocated_res[][],int max_res[][],int avl1[])
    {
        int num = avl1.length ;
        int sa = allocated_res.length;
        int i,j,k;
        int flag=0;
         for(i=0;i<sa;i++)
         {
              for(j=0;j<num;j++)
                  {  if(max_res[i][j]-allocated_res[i][j]>avl1[j])
                      {
                          flag=1;
                          //printf("show %d",i+1);
                          return i+1;
                          //break;
                       }

              }
•                 for(k=0;k<num;k++)
•                 {
•                     avl1[k]=allocated_res[i][k]+avl1[k];
•                 }

•              }
•           return flag;

•       }

•         public static void main(String args[])
•           {
•               BankersAlgo b=new BankersAlgo();


•                                                     int avl[]={3,1,1,2};
•                                                    //int sizeavl=4;
•                                                     int ar[][]={ {1,2,2,1},
•                                                                  {1,1,3,3},
•                                                                  {1,1,1,0},
•                                                                  {3,2,1,1}};
•  int mr[][]={ {3,3,2,2},
•                                       {1,7,3,4},
•                                       {1,1,5,0},
•                                       {2,2,3,3}};

•                               int res=b.getState(ar,mr,avl);

•               if(res==0)
•               {
•                 System.out.println("the state is safe");

•                 }

•               else
•               {
•                 System.out.println("the process no " +res+" fails");
•                }

•      }

•  }
```